Reconstruction Error Characterization and Control: A Sampling Theory Approach

Raghu Machiraju, *Member, IEEE Computer Society*, and Roni Yagel, *Member, IEEE Computer Society*

Abstract—Reconstruction is prerequisite whenever a discrete signal needs to be resampled as a result of transformation such as texture mapping, image manipulation, volume slicing, and rendering. We present a new method for the characterization and measurement of reconstruction error in spatial domain. Our method uses the Classical Shannon's Sampling Theorem as a basis to develop error bounds. We use this formulation to provide, for the first time, an efficient way to guarantee an error bound at every point by varying the size of the reconstruction filter. We go further to support position-adaptive reconstruction and data-adaptive reconstruction which adjust filter size to the location of reconstruction point and to the data values in its vicinity. We demonstrate the effectiveness of our methods with 1D signals, 2D signals (images), and 3D signals (volumes).

1 Introduction

Reconstruction is the process of recovering a continuous function from a set of samples. It is one of the fundamental operations in computer graphics and imaging. Many algorithms, such as texture mapping, image registration, image transformation (e.g., rotation, scaling), and volume rendering, transform a raster (2D or 3D) from a *source space* to a *target space*. All these algorithms must reconstruct the underlying function in either space. Given the essential nature of the reconstruction operation, it is surprising that, although much work has been expended in the design of reconstruction filters, not much attention has been paid to characterize and control its numerical accuracy. Inaccurate reconstruction can manifest in image artifacts and make subsequent operations error prone.

The work described here is aimed to give the user, for the first time, the ability to set a *point-wise* error bound. Unlike existing methods, which use frequency domain analysis to guarantee some global error bound, we use spatial domain error analysis to guarantee that, for a given threshold ε, the difference between the reconstructed function and the real function is not more than ε at *any point*. Our spatial domain analysis culminates in a formal expression for the error bound at every point (15). Examining this expression, we observe a dependency between error magnitude and the location of reconstruction and data values. Unlike existing methods, we can, therefore, adapt filter size to both reconstruction location and data complexity, using rigorous estimates. A shorter version of this work is available in [22]. We present several more results in this paper. In Section 2, we introduce the terminology and methods currently emphasized in reconstruction methods. In Section 3, we describe our approach, and in Sections 4-6, we present our results.

• *R. Machiraju is with the Department of Computer Science and NSF Engineering Research Center for Computational Field Simulation, Box 9637, Mississippi State University, Mississippi State, MS 39762. E-mail: raghu@ERC.MsState.edu.*
• *R. Yagel is with the Department of Computer and Information Science, the Advanced Computing Center for the Arts and Design, Ohio State University, 395 Dreese Lab, 2015 Neil Ave. Columbus, OH 43210-1277. E-mail: yagel@cis.ohio-state.edu.*

For information on obtaining reprints of this article, please send e-mail to: transvcg@computer.org, and reference IEEECS Log Number V96027.

2 Background

An image or a volume is usually in the form of a regular rectilinear *grid* or a *mesh* of sampled function values termed *pixels* (image) or *voxels* (volume). When 2D images are subjected to affine transformations (e.g., translation, scaling, rotation [5], [12], [28]), or when they are subjected to nonaffine grid deformation (perspective, texture mapping [13], warping [2], [38]), the function value in the form of pixel intensity has to be reconstructed on the target grid, commonly called the *resampling grid*. Similarly, resampling is also needed when a 3D volume [17] is subjected to affine transformations (e.g., translation, scaling, rotation [12], or orthographic ray casting [39]) or nonaffine transformations (e.g., perspective ray casting [16], deformation [19]). Voxel intensity, opacity, or color need to be determined at intermediate points inside the volume. Interpolation is the reconstruction method of choice in all the aforementioned algorithms. Reconstruction in the source grid is much rarer and is conducted through the use of reconstruction kernels. Shearing [12] and splatting [37] are the only two well known volume rendering algorithms to reconstruct functions in target grid. However, both forms of reconstruction are equivalent, and, in this paper, we shall examine reconstruction in terms of function interpolation.

In this paper, we distinguish between resampling schemes and reconstruction operations. Resampling schemes provide the points where the functions are reconstructed. The subsequent use of these points is also dictated by the resampling schemes. A typical resampling scheme (e.g., volumetric ray-casting) can be described by the functional $x_k = x_0 + bk$, where x_0 is the location of the first point and b is the distance between the resampling points (Fig. 1a). The signal is thus resampled onto another grid, and the number of data points can change as a result of scaling inherent to the resampling operations. Once the location of the resampling point R is obtained (see Fig. 1b), reconstruction is performed to obtain the function value at R from the known function values S_i. This distinction between resampling scheme and reconstruction is rooted in practice and is different from the purely theoretical perspective that dictates that the function be first reconstructed everywhere and that resampling be subsequently conducted. We now examine some ideal reconstruction methods.

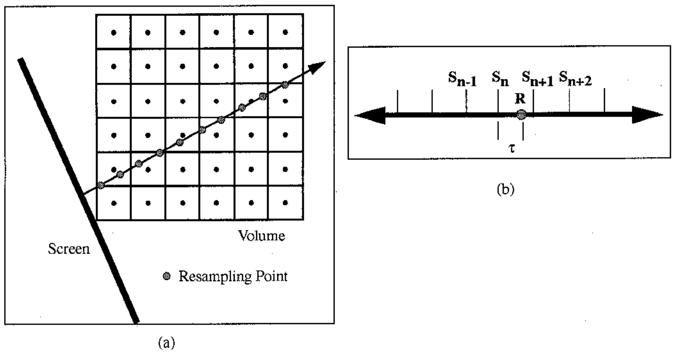


Fig. 1. Difference between resampling and reconstruction. (a) Resampling dictates the set of (gray) points for volumetric ray-casting (schematically represented by a 2D grid). Function values are reconstructed at such intermediate locations. (b) A 1D example illustrating reconstruction at point R from the samples S_i.

2.1 Ideal Reconstruction—Method and Assumptions

Much has been written about the reconstruction of sampled datasets in the fields of signal processing [26], image processing [7], [15], and graphics [10], [38]. We briefly discuss some of the important assumptions and results from this body of literature. Another body of work on the same problem is available in the applied mathematics literature [33], [41], where the process is usually referred to as *interpolation*. Although the following discussion is in terms of 1D signals, it is also applicable to 2D and 3D signals. The results of the following discussion are indeed extended to 2D (in Section 5) and to 3D (in Section 6).

We denote by $f(x)$ a continuous function (the *signal*) which is sampled into the discrete function $f_s(kh)$, where h is an equidistant gap between samples and k is an integer. In many computer graphics, image processing, and scientific visualization applications $f(x)$ is not available; we have only f_s, which is the discrete image we need to manipulate.

The fundamental assumption made in this paper is that the original continuous function, $f(x)$, is *bandlimited*. A function is bandlimited if there exists a frequency ω_c, called the *cut-off frequency*, such that the strength of any frequency component greater than ω_c is zero. These assumptions are not too restrictive, because the notion of bandlimitedness is general and can be applied to many forms of sampled data used in computer graphics and scientific visualization. During the process of acquiring digital images, acquisition devices (e.g., cameras, scanners) perform a filtering operation and bandlimit the function. Images generated by numerical simulations of physical phenomena (common in disciplines such as computational fluid dynamics) are also bandlimited, because typically robust numerical solutions can be obtained only if the algorithm incorporates a smoothing step. Finally, all rendering and scan-conversion algorithms, in order to provide antialiased images, typically employ a filtering step that bandlimits the image. To illustrate this point we examine the frequency spectrum of the mandrill image (Fig. 2). One would expect the frequency content to be very high, given the profusion of very small details. It can be readily seen that the Fourier transform is sparse and can be thought to be limited to a small region around the center. Malzbender presents similar observations for volumes obtained through medical acquisition devices (e.g., CAT Scans, MRI) [20].

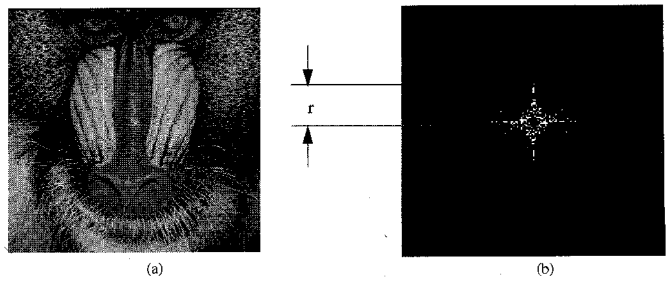


Fig. 2. (a) The mandrill image. (b) The frequency spectrum of the image obtained through a Fourier transform of the original image. It is worth noting that the spectrum is sparse and is limited to a region near the center. The size of the significant spectrum along the Y-axis is measured by the frequency guard r (Section 3.1).

Another important assumption is that the continuous signal f is sampled at or above the *Nyquist frequency*. The Nyquist frequency of a signal is defined as twice the maximum frequency of the signal. Thus, in our context of bandlimited functions, the Nyquist frequency ω_n is given by $2\omega_c$, and the sampling frequency ω_s is always greater than or equal to ω_n. This assumption is essential if we are to reconstruct the function exactly. The Shannon's Sampling theorem states that any bandlimited continuous signal $f(x)$, if sampled at or above its Nyquist frequency (yielding the discrete function f_s), can be reconstructed as shown in (1) (yielding the continuous function f_r) [26], [33]. A final assumption is that $f_s(x)$ is uniformly sampled from $f(x)$.

Thus, we reconstruct with the formula

$$f_r(x) = \sum_{k=-\infty}^{\infty} S(x,k,h) f_s(kh) \tag{1}$$

where

$$S(x,k,h) = \begin{cases} 1 & x = kh \\ \dfrac{\sin \frac{\pi}{h}(x-kh)}{\frac{\pi}{h}(x-kh)} & x \neq kh \end{cases} \tag{2}$$

The function $S(x, k, h)$ is called the *Sinc* function. Reconstruction is essentially a convolution operation between the *Sinc* function and the sampled dataset. The *Sinc* function is an ideal filter. It does not attenuate any of the inherent frequencies in a signal.

We can extend this discussion to multiple dimensions. We use *separable filters*, which sample the data successively along each axis. Thus, in 2D the reconstruction equation becomes:

$$f_r(x,y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} S(x,i,h)S(y,j,h)f_S(ih,jh) \quad (3)$$

For 2D images, the reconstruction requires the product of two *Sinc* functions, and similarly we can use three *Sinc* functions in 3D. Although separable filters are fast and easily implemented, separability typically introduces anisotropic effects because the 2D or 3D separable filter is aligned with the principal axes [24]. Anisotropic effects are always present unless a radially symmetric filter is employed. It is important to note here that the Gaussian is both a separable and an isotropic filter and does not introduce anisotropic errors. In this paper we limit ourselves to separable filters.

2.2 Practical Reconstruction Methods and Errors

The ideal reconstruction process from (1), although realizable, is rarely used in practice, because it requires the contribution from all elements of the dataset. An obvious solution is to truncate the function to include only 2*M* + 1 integer valued points:

$$f_{rt}(x,M,h) = \sum_{k=-M}^{M} S(x,k,h)f_S(kh) \quad (4)$$

The resulting interpolation filter is called a *finite impulse-response* (FIR) filter of order 2*M* + 1 and *linear phase.* The quantity *M* is usually referred to as the half-filter length. We call the ensuing error *truncation error* and it is denoted by e_t.

$$e_t(f_S,x,M,h) = \left|f_r(x,h) - f_{rt}(x,M,h)\right|$$
$$e_t(f_S,x,M,h) = \left|\sum_{|k|>m} S(x,k,h)f_S(kh)\right| \quad (5)$$

Elsewhere in the literature, this error is often referred to as post-aliasing [24]. Truncation error manifests itself as *blurring*, *aliasing* (*jaggies*), and *ringing* in an image. In practice, the function is never reconstructed with a truncated *Sinc*. Truncation is tantamount to multiplying the infinite filter with a spatially limited *rectangular window*. As a result, the frequency spectrum of the truncated *Sinc* filter suffers from distortions in the form of aliasing and oscillations. The oscillations, a consequence of *Gibbs phenomenon* in the frequency domain, manifest visually as an annoying ringing artifact [26].

A common solution is to use a *window function* besides a rectangle [23], [26]. We employ the Hamming window in the work reported here (6). The significant aspect of this function is that it falls gradually to zero at the corners of the window and hence reduces the impact of ringing caused through the use of the rectangular window.

$$w_H(x) = \begin{cases} 0.54 + 0.46\cos\left(\dfrac{\pi x}{M+1}\right) & |x| < M+1 \\ 0 & \textit{otherwise} \end{cases} \quad (6)$$

An appropriate filter can be obtained by multiplying this function with the *Sinc* to obtain a modified *Sinc* function. We can also use any of the commonly used reconstruction filters [36]. Other solutions include using trilinear interpolation (cone filter) and the cubic convolution filters [10], [24]. For multiple dimensions ,we can use a product of two 1D window functions to get a 2D window function.

If the *NS* (Non Sinc) filter is different from the *Sinc* function (and is the windowed *Sinc* filter), then the reconstructed function is given by:

$$\tilde{f}_{rt}(x,M,h) = \sum_{k=-M}^{M} NS(x,k,h)f_S(kh) \quad (7)$$

Thus the total reconstruction error, denoted by e_r, is equal to:

$$e_r(f_S,x,M,h) = \left|f_r(x,h) - \tilde{f}_{rt}(x,M,h)\right| \quad (8)$$

The total error can be divided into two components:

$$e_r(f_S,x,M,h) = \left|f_r(x,h) - f_{rt}(x,M,h) + f_{rt}(x,M,h) - \tilde{f}_{rt}(x,M,h)\right|$$
$$e_r(f_S,x,M,h) \le \left|f_r(x,h) - f_{rt}(x,M,h)\right| + \left|f_{rt}(x,M,h) - \tilde{f}_{rt}(x,M,h)\right| \quad (9)$$

The first component is the truncation error e_t, while the second error arises from the use of a filter besides a *Sinc* function. We call the latter *non-Sinc error,* denoted by e_{ns}. Thus, the total reconstruction error is

$$e_r(f_S,x,M,h) \le e_t(f_S,x,M,h) + e_{ns}(f_S,x,M,h) \quad (10)$$

In [34], a similar characterization of the error was attempted; the reconstruction error was the sum of again the truncation error and errors that arise when filters different from the ideal are used. An application of this error characterization leads to the conclusion that reconstruction with functions different from a truncated *Sinc* leads to lower quality images. However, it is well known that the truncated *Sinc* filter actually possess a lower mean square error from the ideal frequency response than filters obtained through the use of a nonrectangular window [26]. Therefore, it is important to note that this characterization of the reconstruction error is purely numerical and is not based on perceptual considerations that address the issue of the suitability of an image to a human observer. Although there is a positive correlation between numerical error and image quality, not much is known about their relationship [32]. A perceptual model, for example, will assign a larger error to truncated *Sinc* filters than to windowed *Sinc*. We do not even attempt to address these very complex questions here.

2.3 Previous Work

The study of reconstruction errors has received some attention in the graphics and image processing literature. However, in 1D digital signal processing considerable effort has been expended on multirate filtering [6] which can be applied to general function reconstruction to some extent. A sampled signal can be decimated (reduced number of samples) by a factor of *N* : 1 or interpolated (larger number of samples) by a factor of 1 : *M*. By choosing a different set of integers for *M* and *N* different resampled points are obtained. The approach disallows getting all possible resampling points of the signal and also it is conducted on the signal in its entirety. Although multirate filtering may be employed in some situations like supersampling in a raycasting algorithm, it is unlikely to be of much benefit in a general function reconstruction framework. Recently, fractional interpolatory methods have been reported for audio applications [9]. The reported method is similar to the one reported in here, except a pointwise error metric is not employed and once again the focus is on reconstructing a signal in its entirety.

. We now review some previous work from imaging and volume rendering literature. In [29], Parker et al. compared the effectiveness of some reconstruction filters. However, they propose no metrics that can be used to judge the goodness of a reconstruction filter. Mitchell and Netravalli [24] first introduced the reconstruction metrics *pre-aliasing* and *post-aliasing*. Marschner and Lobb [23] further characterized post-aliasing artifacts; the proposed metrics, *smoothing* and *post-aliasing*, can help design a suitable filter. However, none of these measures are well suited for determining the accuracy of an interpolation or filtering scheme on a sampled dataset.

Many filter design methods lie in either the frequency domain or the spatial domain. For example, in the image processing and graphics literature, Keys [18], Max [21], Park and Schowengerdt [27], and Mitchell and Netravalli [24] use spatial methods to design reconstruction filters that satisfy certain functional properties (the existence of derivatives, etc.). All of these methods make assumptions about the interpolated function, and they deliver filters which perform well on smooth functions. On the other hand, Carlbom [3] uses frequency methods to design filters that are solutions of a nonlinear optimization process. This approach yields a filter of finite length whose frequency response closely approximates the ideal filter response.

Among these efforts only Carlbom [3] considers reconstruction error. However, this error is defined in the frequency domain and measures only the deviation of the frequency spectrum from the ideal spectrum (a box in frequency domain). Also, this metric is global in nature and does not provide error control on a point-wise basis. Moreover, this and other global frequency domain methods are not conducive to our goal of adapting the filter size to the resampling location and to local data characteristics.

Our method estimates the filter size for a given resampling location so we can interpolate to a desired level of accuracy efficiently. For example, a less expensive interpolation scheme can be used at some locations (e.g., near grid locations in source space). Similarly, a more expensive scheme is warranted at other locations (e.g., far from grid locations). In addition, we also determine the filter size from the complexity of the data at the resampling point. This gives us an efficient yet accurate resampling method.

There is a body of work dedicated to filtering in texture spaces that attempts to address the issues of error control and adaptivity. Fournier and Fiume [8] use spatial methods and a least square error (L^2 norm) estimate (with data included) to guide efficient and accurate anti-aliasing of textures. They also allow adaptive filtering in a manner similar to *MIP maps*. However, their method is complicated and does not use memory efficiently. Norton et al. [25] use a frequency domain approach to perform adaptive filtering. They use a coarse measure of goodness to clamp all frequencies beyond a certain range. However, the adaptivity is not driven by any user-defined error threshold but is guided by an ad hoc measure of the robustness of the filtering operation in the frequency domain.

Adaptive filtering in the frequency domain is also reported by Totsuka and Levoy [35] for 3D volumes. Again, the adaptivity is not driven by an error threshold. Also, the technique requires that the filtering be conducted in the frequency domain, which requires transforming and storing a a large 3D dataset.

In summary, many of the past efforts provide neither any means of controlling the reconstruction error nor adaptive reconstruction of the continuous signal. The methods that provide error specification and control in the spatial domain are rather complex, while the methods that provide adaptivity are either inconvenient or available in the frequency domain only. These shortcomings motivate our attempt to develop spatial domain methods that allow the specification of error bounds and allow the use of different filter lengths adapted to the reconstruction operation and local data complexity. Our method can be successfully employed in resampling operations and texture mapping. In the next section we provide the necessary theory and develop adaptive spatial-domain methods.

3 Reconstruction Error Estimates

In this section we provide estimates of *truncation* error and the *non-Sinc* error. Researchers in mathematics and electrical engineering have been long concerned with the accuracy of sampling schemes. In both fields, some work has been done that can be used to estimate the truncation error. We present some appropriate results from complex analysis for the sake of completeness. Later, we obtain bounds for the truncation error. The estimates for *non-Sinc* error, which are much easier to obtain, are also presented in this section.

3.1 Truncation Error Estimates

Equation (1) is the starting point for our effort. The true error can be computed for spatially limited signals such as images and volumes. However, in the presence of a large number of sampled data points, the computation can be prohibitively expensive. From (1), we can compute the function value at the resampling point $\boldsymbol{R}$ as follows:

$$f_r(x) = \sum_{k=-\infty}^{\infty} \frac{\sin\frac{\pi}{h}(x-kh)}{\frac{\pi}{h}(x-kh)} f_s(kh) \tag{11}$$

Let the resampling point $\boldsymbol{R}$ lie in cell n, i.e., between sampled data points S_n and S_{n+1}. Also, let τ be the distance of $\boldsymbol{R}$ from S_n; in other words, $x = nh + \tau$. The truncation error is obtained from (11) by dropping $2M + 1$ terms of the above infinite summation. Thus, the truncation error, $e_t(f_s, n, \tau, h, M)$, is given in (12).

$$e_t(f_s, n, \tau, h, M) = f_r(nh+\tau) - \sum_{k=-M}^{M} \frac{\sin\frac{\pi}{h}(nh+\tau-kh)}{\frac{\pi}{h}(nh+\tau-kh)} f_s(kh) \tag{12}$$

A change of variable (let $m = n - k$) allows us to rewrite the above equation in a more convenient form as shown in (13).

$$e_t(f_s, n, \tau, h, M) = \sum_{|m|>M} \frac{\sin\frac{\pi}{h}(mh+\tau)}{\frac{\pi}{h}(mh+\tau)} f_s((n-m)h) \tag{13}$$

Expanding the sinusoidal term inside the summation and letting $sin \pi m = 0$ and $cos \pi m = -1^m$ for all values of m, we get (14).

$$e_t(f_s, n, \tau, h, M) = \frac{\sin\frac{\pi\tau}{h}}{\frac{\pi}{h}} \sum_{|m|>M} \frac{(-1)^m}{(mh+\tau)} f_s((n-m)h) \tag{14}$$

It is easy to determine a bound from (14):

$$e_t(f_s, n, \tau, h, M) \le \frac{h\left|\sin\frac{\pi\tau}{h}\right|}{\pi} \sum_{|m|>M} \frac{|f_s((n-m)h)|}{|(mh+\tau)|} \tag{15}$$

We now make two important observations from (15).

OBSERVATION 1. The truncation error depends on the location of the resampling point R. If R is located at the center of a grid cell in the source space, i.e., $\tau = 0.5h$, it attains its maximum value and drops off to zero as one moves closer to the sampled data locations ($\tau = 0$ or $\tau = h$). The denominator in each term measures the distance of the resampling point from the sampled data. Examining the sum in (15), one finds that it can be split further into two more sums. The denominator of terms in the first sum (for negative values of m) can be rewritten as $-((m-1)h + (h-\tau))$, while in the second sum (for positive values of m) the original expression in terms of m and τ remains. It is now easy to see that the error is maximum for $\tau = 0.5h$, since equidistant samples are given the same weights. In [29], similar observations are made. However, these observations were made in the frequency domain. Also, the error was not quantified. An important implication from this observation is that one can use filters of different lengths depending on the location of the resampling point. In the following section, we shall provide evidence to illustrate this fact using 1D and multidimensional examples.

OBSERVATION 2. For large values of m, the contributions by sampled data values to the error bound is small. In (14), the term which contributes numerically for a given τ is the term $\frac{1}{mh+\tau}$. Using Taylor Series Expansion one could write

$$\frac{1}{mh+\tau} = \begin{cases} \frac{1}{mh} - \frac{\tau}{(mh)^2} + R(m,h,\tau) & m \ne 0 \\ \frac{1}{\tau} & m = 0 \end{cases} \tag{16}$$

where the remainder term $R(m, h, \tau)$ is defined as

$$R(m,h,\tau) = \begin{cases} \frac{\tau^2}{(mh)^2(mh+\tau)} & m \ne 0 \\ 0 & m = 0 \end{cases} \tag{17}$$

The function $R(m, h, \tau)$ is a rapidly decreasing function in m. It therefore suffices to evaluate the function for only a few values of m around the resampling point R. Indeed it has been shown in [34] that it suffices to consider only values of m, such that $|m| < 3$ for a maximum normalized root mean square error of 2.75 percent.

The main use of this observation is the behavior of this bound in two and three dimensions. For multidimensional separable filters, the contributions are even much smaller since each denominator is the product of two or three distance terms. The implication of this observation is that we can effectively limit ourselves to reasonably sized neighborhoods. Thus, for $|m| < 3$, we incur an error of 0.075 percent when we consider truncated sum in *two*-dimensional square neighborhoods each with side of length three pixel units and an error 2.7×10^{-05} percent for three dimensions. This observation allows even more efficient implementations of the reconstruction operation because even smaller length filters can be used.

This error bound in (15) is still not computationally practical because all sampled data points have to be considered. Moreover, this bound overestimates the error because it does not take into account the oscillating nature of the *Sinc* function. We now consider some tractable error bounds that can be used in practice. The error bound for this infinite sum can be found by resorting to complex analysis [40]. However, before we state the relevant results, we discuss the important idea of *frequency guards.*

Frequency guard bands allow the approximation of the infinite sum in (15) by the integral of an analytic function that exists on the real line. A frequency guard of width r, $0 < r < 1$ measures the size of significant spectrum of the signal. Thus, by choosing a guard of size r, we are limiting ourselves to frequencies less than $r\omega_c$. The frequency guard can be found by determining the ratio of the maximum significant frequency of the spectrum and the cutoff frequency. However, for most graphics and imaging applications, it is sufficient to use very crude estimates. In Fig. 2b, the size of the frequency guard along the Y direction is shown. We shall address this issue further in Section 4, in which we discuss results and implementation issues.

The methodology used in [14], [40] can be applied to determine the error of any polynomial approximation scheme. In fact, such a methodology has been used to estimate the error of Legendre and Hermite Polynomial interpolation. The mainstays of this approach are the theory of analytic functions and the application of *Cauchy's Integral Formula and Residue Theorem* [31].

3.1.1 Results from Complex Analysis

We state some results from complex analysis but we do not present proofs here. The Appendix contains the proof. The error bound can be obtained in terms of either the maximum of the function value Max_f, or the spectral energy of the function, E_f. Because all signals under consideration have finite energy and are real and available as sampled datasets, E_f can be simply determined by:

$$E_f = \sqrt{\frac{h\sum_k |f_s(k)|^2}{2\pi}} \tag{18}$$

This relationship is a direct consequence of the Cardinal Series Expansion [33]. We now state Theorem 1, which expresses the truncation error bound in terms of the spectral energy of the function.

THEOREM 1. *The truncation error $e_t(f_s, x, k, h, M)$, in terms of the total energy of the signal, is bounded from above by the quantity*

$$e_t(f_s, x, k, h, M) \le \frac{2E_f\sqrt{\frac{r}{h}}\left|\sin\frac{\pi x}{h}\right|}{\pi^2(1-r)M} \tag{19}$$

Thus, we are able to express the truncation error bound in terms of the energy of a function and the frequency guard r. If once again $x = nh + \tau$, i.e., it lies in the cell n of the source grid, we can replace x with τ (refer to (15)). We now state another theorem, Theorem 2, which expresses the error bound in terms of the maximum of a function. The proof for this theorem is similar to that of Theorem 1 and we are therefore not including it, for the sake of brevity.

THEOREM 2. *The truncation error $e_t(f_s, x, k, h, M)$, in terms of the maximum value of a function, Max_f is bounded from above by the quantity*

$$e_t(f_s, x, k, h, M) \le \frac{Max_f\left|\sin\frac{\pi x}{h}\right|}{\pi M\cos\frac{r\pi}{2}} \tag{20}$$

We now characterize the error that arises from the use of a function different from a *Sinc* function.

3.2 Non-*Sinc* Error

The use of the truncated *Sinc* induces visual artifacts including ringing, blurring, and aliasing. Therefore, other interpolating functions are used including either a specially designed one (e.g., cubic convolution) or the *Sinc* function suitably modulated by a smooth window function. We, however, need to estimate the error that arises from the use of windowed function. Thus, we now have a function *NS* instead of the *Sinc* function *S*. Once again we can use either the spectral energy or the maximum value of the function. Using Parseval's Theorem [26] and (18), we can write

$$e_s(f_s, x, n, h, M) \le E_f(M)\sqrt{\frac{1}{2\pi}\int_{-M}^{M}\left|S(t,0,h) - NS(t,0,h)\right|^2 dt} \tag{21}$$

The integral computes the difference between the two functions in the L^2 norm space. The quantity $E_f(M)$ is the energy of the signal in a $2M + 1$ sized neighborhood around the resampling point $\mathbf{R}$. Because the filters are space invariant we evaluate the filters when placed at $x = 0$ for sake of convenience. One can similarly define a bound including the maximum value of the function:

$$e_s(f_s, x, n, h, M) \le Max_f(M)\int_{-M}^{M}\left|S(t,0,h) - NS(t,0,h)\right|dt \tag{22}$$

The quantity $Max_f(M)$ is the maximum value of the function in a $2M + 1$ neighborhood. Once again, we are determining the difference in the areas of the two filter functions. The *non-Sinc* error is much smaller in value in comparison to truncation error. We measured the error using (21) and (22) and compared the values against corresponding estimates of truncation error. In the case of windowed *Sinc* it was noted that the *non-Sinc* error was quite insignificant compared to the truncation error (by a factor of five at least). Therefore, we do not consider it further in this paper. In [34], a similar division of errors was conducted and once again the truncation error was mostly considered.

In this section, we described the errors that arise from filtering operations. In Section 4, we use these measures to predict reconstruction errors that arise from representative resampling operations and then show how these predictions can be used to perform adaptive reconstruction.

4 Accurate and Adaptive Reconstruction of 1D Signals

In this section, we test the validity of the bounds on 1D signals. We also illustrate the usefulness and viability of adaptive schemes. In the subsequent sections we implement our schemes for particular 2D and 3D resampling schemes.

In Fig. 3 we consider a 1D signal obtained from row 300 of the Lenna image. It is worth noting that the signal under consideration has very small energy content. One can estimate the value of the frequency guard, r, by simply computing the first few Fourier coefficients above a user-defined threshold. It was observed that the value of the guard was usually less than 0.1 for all images considered for this work. In other words, most of the energy of the function is characterized by the first one-tenth of the Fourier coefficients. The more accurately one measures the frequency guards, the better the estimates are. However, even coarse estimates can suffice for many signals and resampling situations in computer graphics.

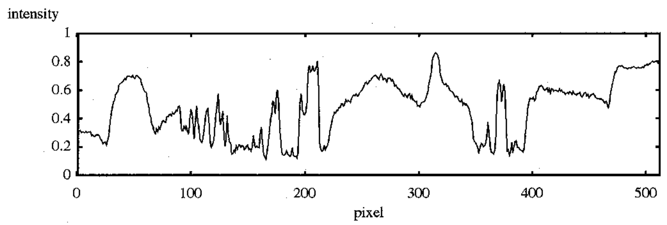


Fig. 3. Intensity values for scanline 300 of the 512x512 *Lenna* image. The intensity values are normalized to the maximum possible pixel value, namely 255.

The actual error from (15) and the error estimates from (19) (using energy) and (20) (using maximum values) are determined when the signal is resampled onto a new grid (Fig. 4). The function is reconstructed at $x_k = x_0 + 0.99*k$, where x_0 is the location of the first row pixel and k is an integer. The estimates from (20) are looser, and we found the energy estimates closer to the actual error for many signals and resampling schemes.

Here we can actually see evidence for Observation 1 made in Section 3.1. The error behaves in a periodic manner for the representative resampling scheme. If a larger resampling frequency is chosen, the periodicity of the error is

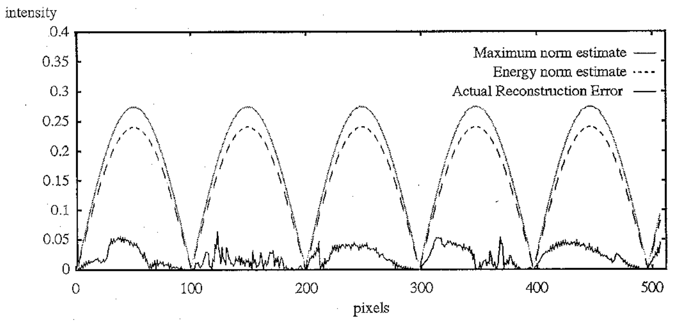


Fig. 4. Comparison of Error Estimates. The actual error and the estimates from (19) (energy) and (20) (maximum) are plotted. Both estimates bound the actual error well, the energy estimates being closer.

higher. One can readily conclude that the filters of the same length need not be used everywhere during the resampling operation. To use different filters at different resampling positions, we can use the error estimates of (19) and (20). For instance, we can set the point-wise error to ε for all x along the length of the signal. The required filter length at resampling point x can be determined from the computation listed in (23).

$$M(x) = \frac{2E_f\sqrt{\frac{r}{h}}\left|\sin\frac{\pi x}{h}\right|}{\pi^2(1-r)\varepsilon} \tag{23}$$

The dashed line in Fig. 6 shows the minimum filter length at all points required for the resampling of the signal of Fig. 3 to obtain a user defined accuracy of $\varepsilon = 0.02$. The maximum filter length employed for reconstruction is 27 (= 13*2 + 1). We call this filtering scheme position-adaptive, because the size of the filter is influenced only by the position of the resampling point. If frequency domain methods of filter design are used, usually filters of even greater length are obtained from the design process. Also, it is not certain that the desired level of accuracy is guaranteed from the application of such a filter.

In Fig. 4, we used the total energy or the global maximum value of the function to compute the bounds. The estimated bounds are conservative. Taking into account the rapid decay of the *Sinc* function as one moves away from the resampling point, it might be useful to consider the energy or maximum of a function over a neighborhood of somewhat significant size as stated in Observation 2 of Section 3.1. The problem is now reduced to determining a window of appropriate size that is suitable for a given signal. This can be determined easily from the estimates of the bounds itself. We can set the minimum error of resampling ε_{min} that can possibly arise during resampling. Then we can simply calculate the neighborhood size M_e by using either (19) or (20). We use energy estimates to determine the optimum neighborhood size.

$$M_e = \frac{2E_f\sqrt{\frac{r}{h}}\left|\sin\frac{\pi}{2h}\right|}{\pi^2(1-r)\varepsilon_{min}} \tag{24}$$

The value of τ is set 0.5 to cover all possible resampling positions. Now we can determine the maximum or the energy over a neighborhood of this size. Equation (23) then becomes

$$M(x) = \frac{2E_f\sqrt{\frac{r(M_e)}{h}}\left|\sin\frac{\pi\tau}{h}\right|}{\pi^2(1-r)\varepsilon} \tag{25}$$

A preprocessing step is now required that computes the energy or the maximum of the function value over neighborhoods. In Fig. 5, we plot the true error and the estimates using energy and maximum values over a neighborhood of size $M_e = 25$. The estimate based on energy is now very close to the actual error. The solid line in Fig. 6 shows the sizes of the filters used when a neighborhood of size 25 is used for the signal of Fig. 3. The maximum and average size of the filters are significantly lowered. The use of smaller neighborhoods yields smaller filters and hence savings in reconstruction time. We call this filtering scheme *data-adaptive*.

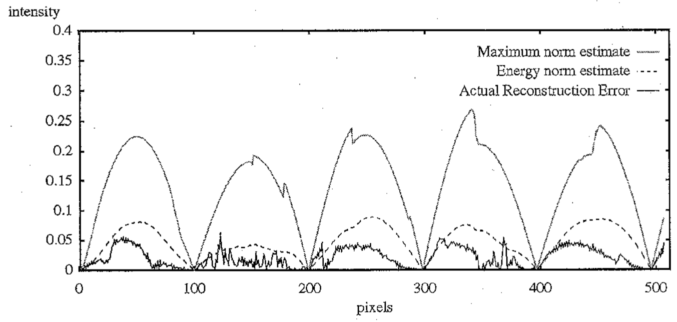


Fig. 5. Comparison of Error Estimates. The actual error and the estimates from (19) (energy) and (20) (maximum) are plotted when a neighborhood of size 25 is used to compute the maximum or the energy. The estimate based on energy follows the actual error very closely.

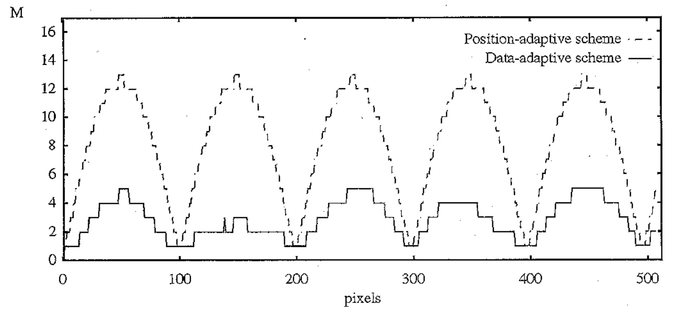


Fig. 6. Comparison of Filter Lengths. The filter half length, M, is plotted for the signal of Fig. 3. The dashed curve provides the length of the filter at the resampling points when the position-adaptive scheme is used. The solid curve provides the filter lengths for the data-adaptive scheme. The lengths were obtained using energy estimates for an error threshold of 0.02.

The behavior of the bounds is different when the same signal is subjected to a plain translation. This form of resampling occurs during shearing transformations, image registration, optimized versions of volumetric ray-casting, etc. The error does not behave in the same periodic manner as before because the displacement along the source space grid is constant. The truncation error is dictated more by the data complexity than by the position of the resampling point. We do not show the error plots for such resampling schemes.

In all our 1D experiments, we used the rectangle window function to compute the truncation error and hence did not incur the *non-Sinc* error. If another window function were employed, the total error would no longer be the same. Although the total error no longer reaches zero, the periodic nature of the resampling error remains unchanged. Although a nonrectangular window is used, the adaptive scheme can still be based on the truncation error. Finally, to provide a basis of comparison we reconstructed the signal of Fig. 3 with

- an infinitely long *Sinc* filter (no truncation);
- a truncated *Sinc* filter whose length is not dependent on the data complexity (position-adaptive);
- a truncated *Sinc* filter whose length is influenced by the data complexity (data-adaptive); and
- a cubic spline filter described in [24].

Since the signals in question are finite in extent, we may use *Sinc* functions whose lengths are limited by the length of the signal. However sufficiently long *Sinc* functions provide accurate enough estimates of truncation error. We then determined the errors of reconstruction by computing the difference between the perfectly reconstructed function (using the infinitely long *Sinc* filter) and the functions reconstructed using the nonadaptive, adaptive, and cubic spline filters. We also set a threshold of 0.02 for both filtering schemes as before. In Fig. 7, we plot the reconstruction errors as measured against the perfectly reconstructed signal. The position-adaptive scheme always delivered reconstruction to the desired level of accuracy (= 0.02). The data-adaptive scheme for most of the signal fared well. However, in regions of rapid changes in function value, it underestimated the error. The cubic convolution scheme, on the other hand, was not sensitive to either the position of the resampling location or the data complexity. The error of reconstruction was also sometimes much larger than the desired level of 0.02. If the desired level of accuracy is reduced to 0.002, both adaptive schemes fare well, while the performance of the cubic convolution filter remains the same. Having shown the effectiveness of our error measures we now provide 2D examples.

5 Accurate and Adaptive Reconstruction of 2D Images

We considered a few two-dimensional images to show the usefulness of the methods developed here. Equations (19) and (20) now simply become

$$e_t(f,x,y,k,h,M) \le \frac{L_1 E_f \frac{\sqrt{r_x r_y}}{h}\left|\sin\frac{\pi\tau_x}{h}\right|\left|\sin\frac{\pi\tau_y}{h}\right|}{\pi^4 M^2 (1-r_x)(1-r_y)} \tag{26}$$

$$e_t(f,x,y,k,h,M) \le \frac{Max_f \sin\pi\frac{x}{h}\sin\pi\frac{y}{h}}{\pi^2 M^2 \cos\frac{r_x\pi}{2}\cos\frac{r_y\pi}{2}} \tag{27}$$

The quantities r_x and r_y are frequency guards for each of the dimensions in the frequency domain and can be crudely estimated from the FFT. The energy and maximum values are now determined for all points in the image. The error estimates and filtering schemes can be adapted to images and volumes very easily. The position-adaptive scheme does not require any preprocessing, while the data-adaptive scheme requires that the energy or the maximum of the underlying function be determined over a neighborhood. By specifying the minimum desired error, we can use a derivative of (27) (and similar to (24)) to determine the size of neighborhood required to achieve the desired error of ε_{min}. The local energy and maximum values are then stored for each pixel. At each resampling point, filter size is then determined by using the error estimates, and then applied in the 2D neighborhood. We also employ the 2D *Hamming* window to obtain images of higher visual quality.

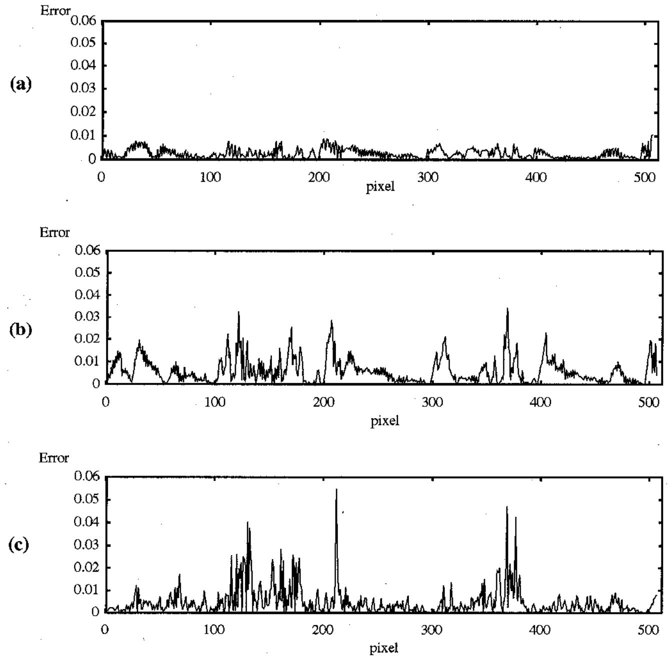


Fig. 7. Effectiveness of Reconstruction Schemes. (a) Error obtained from position-adaptive scheme. (b) Error from data-adaptive scheme. (c) Error from a cubic convolution filter. The filter sizes are shown in Fig. 6. The error threshold was set to 0.02.

Fig. 8a shows the image of a simulated flow rotated by 25 degrees and scaled down by 0.75 along both dimensions while guaranteeing error threshold $\varepsilon = 0.02$. Unlike in the more complicated shearing schemes used in image manipulation packages, we employed a very simple resampling scheme: A bounding box is first found and all pixels within it are scanned and mapped back to the source space of the original image. Fig. 8b provides a comparison between the error in the data-adaptive and the position-adaptive schemes. One can see that differences exist in areas where the pixel intensity changes significantly over small areas, e.g., near edges.

TABLE 1
HALF-SIZE OF FILTERS USED
IN THE RECONSTRUCTION OF FIG. 8.

Method	Minimum	Maximum	Average
Position-Adaptive	1	11	6.75
Data-Adaptive	1	5	2.33
Cubic Convolution	2.5	2.5	2.5

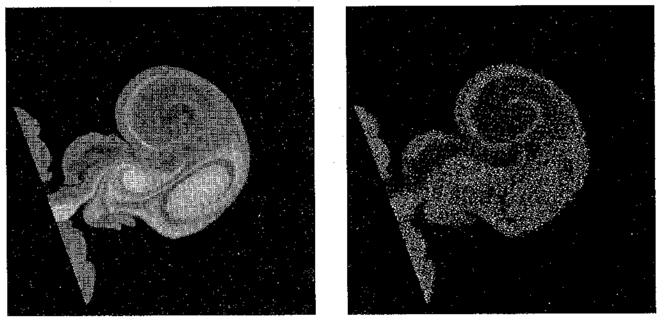


Fig. 8. (a) Flow simulation image after affine transformations of scaling by 0.75 (along both directions) and rotation of 25 degrees. Reconstruction was done with the position-adaptive method with $\varepsilon = 0.02$. (b) A difference image between the position-adaptive and the data-adaptive methods. Differences were exaggerated to make the pattern visible.

Fig. 9 shows, in the form of a gray scale image, the different filter sizes used for the rotation. The difference in the filter lengths at various resampling points is determined and assigned suitable gray-scale values, where white represents larger filter sizes. As evident from Fig. 9a, the filter size changes in a periodic sinusoidal fashion. Also, the filter size adapts to the data complexity, as shown in Fig. 9b. For instance, the dark areas around the flow are reconstructed with smaller length filters. The position of the resampling point still modulates the filter size. Table 1 provides a comparison of the filter sizes for the position-adaptive, data-adaptive, and the traditional cubic convolution filter [24]. Similarly, we show yet another example of image reconstruction to illustrate that the methods work on images of various kinds. Finally, we provide another example to illustrate the usefulness of our methods. In this example the image of the mandrill is subjected to the following transformations:

- translate by (0.3, 0.3) and rotate by 45 degrees;
- translate by (–0.3, –0.3) and rotate by –20 degrees;
- translate by (0.1, 0.1) and rotate by –5 degrees; and
- translate by (–0.1, –0.1) and rotate by –20 degrees.

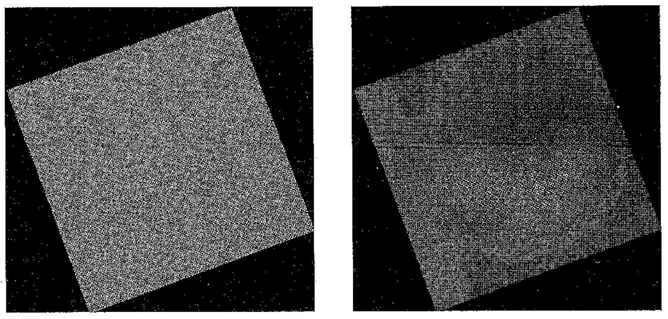


Fig. 9. Filter size used for generating Fig. 8 for position-adaptive filters (a) and for data-adaptive method (b). Bright values stand for larger filter sizes. The error threshold here was set to 0.02. The average filter size was measured to be at 2.4.

Similar repeated resampling schemes have been used by other authors to show the viability of interpolation schemes [3], [29]. Thus, we get our final image back in the position we started out with. By comparing the final image with the initial image, we can estimate the cumulative error incurred. Here we compared the results when the interpolation was performed by bilinear, cubic, or by our adaptive methods. Fig. 10 shows the results. It is easy to notice (near the eyes and mane) that most blurring occurred when the bilinear filter was used, while the least occurred when our adaptive methods were used. A numerical comparison of the final and original images also supports the claim that our methods generate the minimum error.

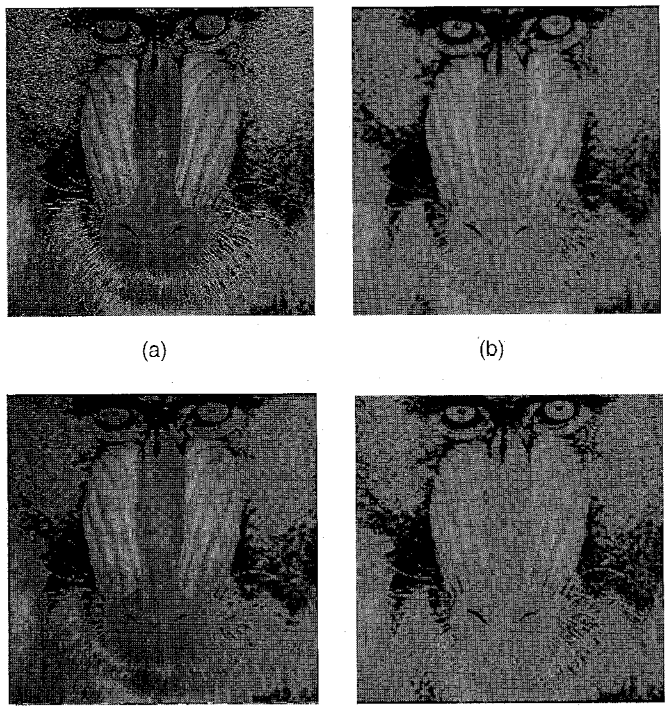


Fig. 10. Comparison of interpolation schemes. (a) Original image. (b) Interpolation with bilinear interpolation (half-filter size 1). (c) Interpolation with cubic convolution (half-filter size 2). (d) Interpolation with accurate and adaptive methods. The error threshold was set to 0.02. The average half size of the filter for the adaptive scheme was measured at 2.6.

6 Accurate and Adaptive Reconstruction of 3D Signals

In this section, we implement our schemes for particular 3D resampling schemes (slicing and volume rendering). We also show the impact of accurate function evaluation by estimating surface normals as part of a simple ray-caster and then using the estimated normal in a shading algorithm.

6.1 Slicing

The error bounds and filtering schemes can also be adapted to volumetric applications like slicing. A position- or data-adaptive scheme similar to the one presented in the previous section is useful to guide the reconstruction process inherent in the slicing algorithm. The data-adaptive scheme requires that the energy or the maximum of the underlying function be determined over a neighborhood. To compute these quantities, we developed efficient algorithms that scan the image or volume in a systematic manner and exploit the inherent coherency of the computations through the use of circular buffers. Once the local energy or maximum is found for each voxel, it is stored in a volume of the same size as the original.

The slicing algorithm rotates the desired slicing plane (described by a direction vector for the normal and an offset from the origin) onto the XY plane thus defining an implicit affine transformation T. The advantage of this slicing implementation is its ability to control the resampling scheme. Another reported work on volumetric slicing is found in [30]. The accuracy of the method reported is limited because trilinear interpolation is used to determine function values at intermediate locations. To use the adaptive schemes of Section 4, we use a derivative of (20) to determine the filter size (28).

$$M^3 = \frac{Max_f \left|\sin\frac{\pi x}{h}\right|\left|\sin\frac{\pi y}{h}\right|\left|\sin\frac{\pi z}{h}\right|}{\pi^3 \varepsilon \cos\frac{r_x\pi}{2}\cos\frac{r_y\pi}{2}\cos\frac{r_z\pi}{2}} \tag{28}$$

Fig. 11 shows slices of a 256^3 volume MRI head dataset reconstructed by our adaptive schemes. The slicing plane in Fig. 11a is almost vertical, while the one in Fig. 11b has a normal with direction cosines of (1, 1, 1) and passes through the center of the volume. The values of frequency guards r_x, r_y, and r_z were found to be 0.086, 0.055, 0.031, respectively. We employ *Hamming* windows to subdue ringing effects for our schemes. The error threshold for the adaptive schemes is 0.01, while the minimum error used to determine the optimal neighborhood size is 10^{-5}. This minimum error threshold translates to a neighborhood size of 31 for the data-adaptive scheme. We also measured the filter sizes for the oblique slicing scheme. The average half filter size for position-adaptive filtering scheme was measured closer to 2 (Fig. 11c), while the same quantity was measured at 1 for the data-adaptive scheme (Fig. 11d). As shown in Fig. 11d, higher order filters are only used at some resampling points. Once again, for the position adaptive scheme, the half filter size varies in a periodic manner. On the other hand, the half filter size varies less regularly throughout the slice. It is important to note that there still exist some regions where the variation in filter size occurs due to changes in data complexity. However, the variation of the data complexity is not significant enough for the regular pattern of the position-adaptive scheme to be completely skewed. This explains the presence of checker patterns in Fig. 11d.

To compare the quality of the reconstruction, a useful visual tool is the gradient image (Fig. 12). The gradient image consists of two components, namely magnitude and angle. The gradient angle image shows the orientation of the gradient and hence surface. We show the gradient magnitude images for the slices obtained by trilinear interpolation (Fig. 12a and b) and by our methods (Fig. 12c and d). Blurring in an image causes thickening of edges and loss of fine detail. This blurring manifests itself in a magnitude image by thicker edges in the gradient magnitude images. For example, the image of Fig. 12a has thicker edges than that in Fig. 12c. The gradient angle image for the trilinear interpolated slice (Fig. 12b) is more blocky and disoriented than the one obtained by our methods (Fig. 12d). Optimization can be realized by precomputing subexpressions (involving the frequency guards and constants) and by using precomputed tables. As a result, the actual time expended in obtaining the slice of size 256^2 (Fig. 12b) using the adaptive schemes (2.6 seconds) is only three times as much as the time expended when the trilinear filter is filter is used (0.74 seconds).

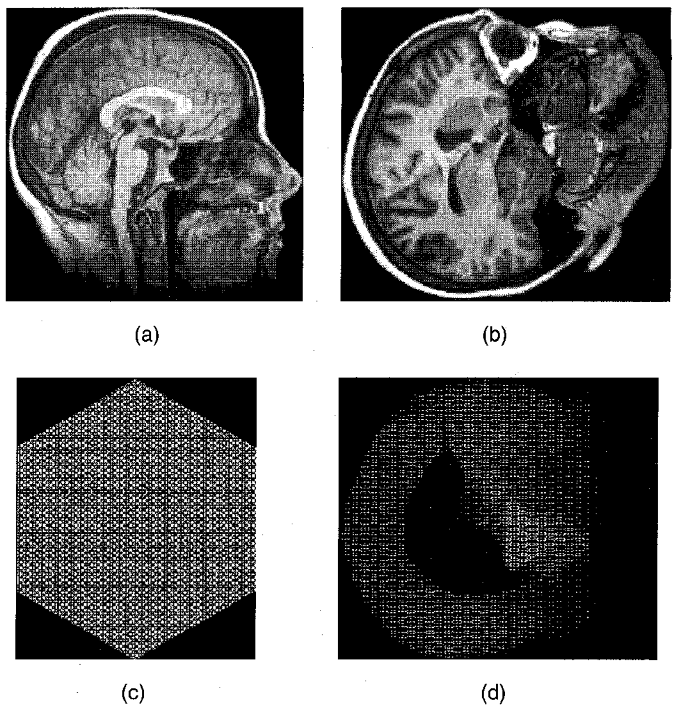


Fig. 11. Oblique slicing of a 3D MRI head dataset. (a) An almost vertical slice passing through the center. (b) An oblique slice through the center with direction cosines (1, 1, 1). (c) The fitler sizes used in the position-adaptive scheme (average half-filter size was measured at 2). (d) Filter sizes used in the data-adaptive scheme (average value was measured at 1). The error threshold was set to 0.01.

6.2 Normal Estimation and Shading

In applications like volumetric ray-casting, the shape of the volume is discerned though shading. Typically, Phong shading is used [16]. The Phong illumination operator P takes the derivative (normal) at a point x, together with light, surface, and eye position information and computes the color at the point x. We denote the derivative of the continuous function f by $\hat{f}$ and the derivative of the reconstructed continuous function f_r by $\hat{f}_r$. The discrete image of $\hat{f}$ is denoted by $\hat{f}_s$. Since, except for the derivative, all other parameters are known, we will denote the illumination operator by $P(\hat{f}_r(x))$.

A cheap way to estimate normals (i.e., differentiate) is through the use of the central derivative operator. Thus, the normal, N, at a point is given by the difference in the function values measured at grid distances along each principal direction. At each of these points, the function value can be determined by any of the interpolating schemes (29).

$$\begin{aligned} Dx &= f(x+h,y,z) - f(x-h,y,z) \\ Dy &= f(x,y+h,z) - f(x,y-h,z) \\ Dz &= f(x,y,z+h) - f(x,y,z-h) \\ N &= (Dx,Dy,Dz)/2h \end{aligned} \tag{29}$$

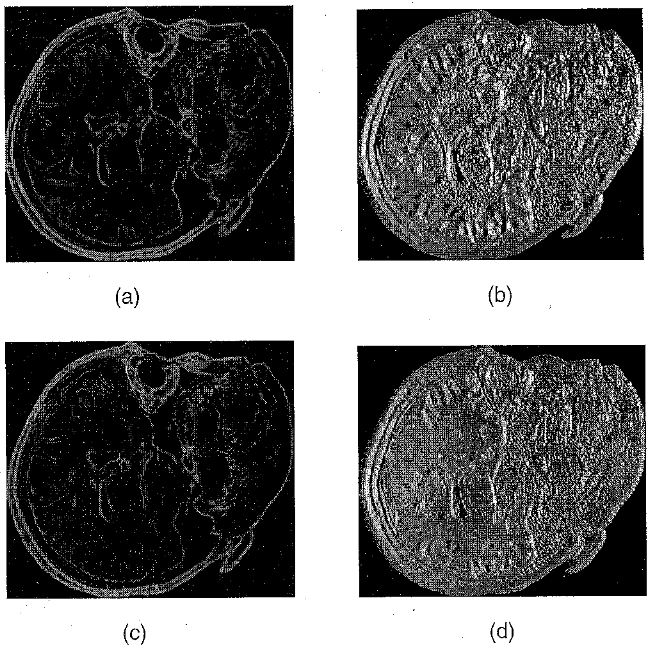


Fig. 12. Comparison of image quality using gradient images. (a) Gradient magnitude image. (b) Gradient angle image of oblique slice obtained by trilinear interpolation. (c) Gradient magnitude and (d) gradient angle image of slice obtained by adaptive methods. The improvement through our methods is obvious.

The central difference operator is far from an ideal high-pass filter since it eliminates fine details (i.e., high frequency data). One can use a better normal estimation method such as the cubic spline filter [1] or a truncated ideal differentiation filter [11]. Nevertheless, the central difference is commonly used for its low computational cost as well as its tendency to suppress noise.

The process of computing $P(\hat{f}_r(x))$, for any point x, can be implemented in several ways. We survey these, starting from the most time consuming method, up to the fastest one.

1) Reconstruct the continuous function f_r from $f_s[kh]$, as in Section 3, by convolving it with a reconstruction filter, and then differentiate it at point x to compute $\hat{f}_r(x)$. Finally, use this value in the shading equation to compute $P(\hat{f}_r(x))$. If we denote by H the reconstruction filter, and by $\otimes$ the convolution operator, then this method can be summarized by the expression:

$$P(\hat{f}_r) = P\left(\frac{d}{dx}(f_s \otimes H)\right) \tag{30}$$

This is a rather expensive approach since reconstruction will have to be performed for each of the differentiation filters (e.g., six points in (29)). One can replace the differentiation in (31) with a convolution by a filter which we denote by D. Therefore, we can write:

$$P(\hat{f}_r) = P(D \otimes (f_s \otimes H)) \tag{31}$$

2) Reconstruct the continuous derivative $\hat{f}_r$ directly from $f_s[kh]$ by convolving it with a specially designed differentiation filter, and then sample it at point x [1], [23]. This approach can be derived directly from (30) by expressing the convolution as a sum and then taking derivative inside the summation.

$$P(\hat{f}_r) = P(f_s \otimes (D \otimes H)) \tag{32}$$

For example, if we use the ideal reconstruction filter, the *Sinc,* to implement H, then the ideal differentiation filter is:

$$D \otimes H \equiv \frac{d}{dx}(\sin(x)/x) = \cos(x)/x - \sin x / x^2 \tag{33}$$

3) Compute the discrete derivative function $\hat{f}_s[kh]$ from $f_s[kh]$. For example, if we use central difference for implementing D, then the (x, y, z) points in (29) are grid points and h is an integer. Since this computation is done only at the grid points, there is no need for reconstruction at this step. Then, one reconstructs $\hat{f}_r$ at the arbitrary point x from $\hat{f}_s[kh]$ by convolving it with a reconstruction filter [11]. Finally, this value is used to compute $P(\hat{f}_r(x))$. This approach is a commutative variation of (31):

$$P(\hat{f}_r) = P((D \otimes f_s) \otimes H) \tag{34}$$

The derivative of the sampled function can be computed because we use only the already available function values, that is grid points, in which case $f_s = f$.

4) As in the previous method, compute $\hat{f}_s[kh]$. Then use these values to compute $P(\hat{f}_s[kh])$, the illumination at the grid points. Then reconstruct the illumination at the point x by convolving it with a reconstruction filter as in Section 3. This approach can be summarized by:

$$P(\hat{f}_r) \approx P(D \otimes f_s) \otimes H \tag{35}$$

Since P is not a linear operator, this approximation is not equivalent to the first three methods.

The first and third approaches are equivalent. However, if one would use perfect reconstruction and differentiation filters, all three methods will indeed result with the same accurate illumination value. The fourth approach is the most efficient method since illumination is not computed for every sample point. For this reason, it is a commonly used method for volume shading [16]. We compare our reconstruction accuracy in the framework of the fourth approach. That is, we shade the data points using Phong shading and then interpolate (reconstruct) the color at the resampling points [16]. In addition to being fastest and most widely used, this method also provides the best way to show the effects of the reconstruction process with no interference from the derivative and illumination operators.

We implemented a simple ray-caster, which shoots horizontal rays into the volume. The results of using trilinear and accurate interpolation schemes on the MRI head dataset are shown in Fig. 13. The rays were terminated when intensity (not opacity) value above a certain threshold was detected. At this point, we employed a preshaded volume to compute the shade which was assigned to a pixel. The function determination for ray termination was performed using a trilinear interpolation scheme. The choice of this interpolation scheme at this stage of the algorithm was dictated more by considerations of efficiency. However, the actual shade computations were computed with adaptive filters for Fig. 13b. We did not include composition in our implementation since it introduces additional phenomenon which require further analysis that is beyond the scope of this work. Our implementation makes it easier to discern the effects caused by reconstruction. Sufficient offsets and a zooming facility allow arbitrary resampling schemes to be tested.

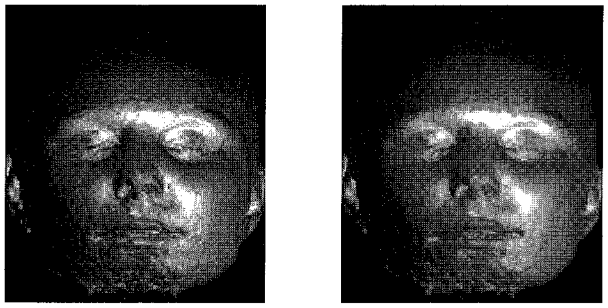

(a) (b)

Fig. 13. Ray-casting example (a) using trilinear interpolation and (b) using accurate methods.

7 Conclusions

We developed a new approach to the characterization and measurement of reconstruction error. Our method, based on spatial domain error analysis, uses approximation theory to develop error bounds for reconstruction. We provide an efficient way to guarantee an error bound at every point by varying filter size. In addition, we support position-adaptive and data-adaptive reconstruction, which adjust filter size to the location of reconstruction and the data complexity. While the position-adaptive scheme adheres better to the error bound, the data adaptive has a smaller average filter size. Performing accurate reconstruction can potentially shift the burden from resampling schemes to reconstruction, thus allowing the use of simpler resampling schemes in many computer graphics applications such as image processing, volume rendering, and texture mapping. Our methods provide the user with a powerful tool for achieving any desired image quality, while incurring space and computation cost that are comparable to those of existing methods. As part of future work, we hope to consider filters other than *Sinc* and windowed *Sinc* which will require a closer look at the *non-Sinc* error and employ other sampling theorems especially those from the wavelet and frames literature.

Appendix: Bounds for Truncation Error

Here we present results from Complex Analysis which are used in the derivation of bounds for the truncation error of the Shannon's Sampling Theorem. Inherent to this approach is the use of a contour *C* or a directional closed path in the complex plane. Such a contour is shown in Fig. 14. Thus, the contour (red counterclockwise path) in our example is a rectangle of size $d \times 2M + 2$ centered at the resampling point ***R***, where d is the height of the contour. The contour is larger than the filter by a distance of unit h (or a distance of $h/2$ on both sides). This size allows for all the $2M + 1$ sampled data points required for interpolation to be inside the contour.

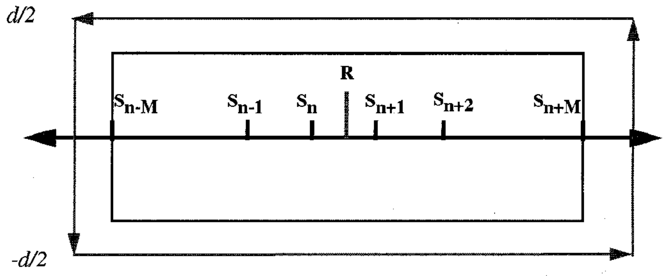


Fig. 14. Contour Integration applied to the computation of error bounds. The inside rectangle is equivalent to the actual filter with half length *M*, while the outside rectangle is the contour ***C***.

The intuition behind using the contour is that it generalizes the process of filtering to the complex plane. The convolution in (15) is embedded in the complex plane. For example, the *Sinc* function, $S(z, h)$, still evaluates to $S(x, h)$ at points $z = x$. Similarly, the underlying function is still defined at the integral values on the *X-axis*. However, on points $z = x + jy$, $y \neq 0$, the reconstructed function and the *Sinc* are evaluated in the complex plane. The contour *C* is the equivalent to the filter used to reconstruct real valued signals and could be of any shape. It is imperative though that it needs to be closed and directional [31]. The rectangle is normally chosen for its simplicity.

The embedding of the infinite sum onto the complex plane enables one to convert the evaluation of the infinite sum into a problem of evaluating an integral over a closed contour. The *Cauchy Residual Theorem* provides the machinery to compute the equivalent integral over the contour. We thus evaluate the integral over a finite sized interval on the *X-axis*. Along the *Y-axis*, however, the region of integration is still unbounded. The key is to show that the equivalent integrand approaches zero as one moves away from the horizontal axis in the vertical direction. More details can be found in [31].

We now state a theorem which provides us with a way to compute the error bound for filtering interpolation schemes. This contour integration used in the theorem once again generalizes the infinite sum of the contributions from sampled data points to the reconstructed value. Further details on contour integration and other results in complex analysis can be found in [31].

THEOREM. *Let **C** be the contour (shown in* Fig. 14) *over the domain **D** (a subset of the complex plane), and let function f be analytic everywhere therein (i.e., it is defined everywhere, and all derivatives exist). Let $G(z) = sin(\pi z/h)$, where z is a point in the complex plane, and let the set Ind(M) of sampled data points lying within the confines of the contour **C** be defined as follows:*

$$Ind(M) = \{k \mid n - M - 1 < k < n + M + 1\} \tag{36}$$

Then, $f_1(z) = f(z)/G(z)$ is analytic everywhere except at $z = kh$, $k \in Ind(M)$, where the function $G(z)$ evaluates to zero. If the function is resampled at $z = x$ on the real line, then the truncation error there is given by

$$e_t(f, x, k, h, M) = \frac{G(x)}{2\pi i} \oint_C \frac{f(z)}{G(z)(z - x)} dz \tag{37}$$

PROOF. The starting point of this proof is the Cauchy Integral Formula [31], which allows us to compute the function $f_1(x)$ as follows:

$$\frac{f(x)}{\sin \pi \frac{x}{h}} = \frac{1}{2\pi i} \oint_C \frac{f(z)}{\sin\left(\pi \frac{z}{h}\right)(z - x)} \tag{38}$$

As mentioned earlier, the integral is performed on a contour C (Fig. 14). To actually evaluate the integral, the contour C is altered because the function $G(z)$ evaluates to zero at all sampled data points, $z = kh$, $k \in Ind(M)$. Thus at these points *poles* are introduced since the function $f_1(z)$ is undefined and the new contour C' now skirts around these points (Fig. 15). In the limiting case, the contour C replaces the newer one C'. The integration over the contour C' is divided as follows:

- the integrals evaluated on clockwise contours around sampled point S_k, $k \in Ind(M)$, each of which is denoted by $Q(S_k)$;
- the integrals along both the straight lines leading to and from the poles; these cancel each other;
- the integrals along the horizontal contours C_1 and C_3; and
- the integrals along the vertical contours C_2 and C_4.

The residues or the quantities $Q(S_k)$ at each one of the poles are evaluated in the limiting case and are given by

$$Q(S_k) = \frac{f(kh)}{\frac{d}{dz}\left(\sin \pi \frac{z}{h}\right)\Big|_{z=kh} (kh - x)} = \frac{(-1)^k f(kh)}{\frac{\pi}{h}(kh - x)} \tag{39}$$

Thus, the right-hand side in (38) can be written as

$$\frac{f(x)}{\sin \pi \frac{x}{h}} = \frac{1}{2\pi i} \oint_{(C_1+C_2+C_3+C_4)} \left(\frac{f(z)}{\sin\left(\pi \frac{z}{h}\right)(z - x)} \right) dz + \sum_{k \in Ind} Q(S_k) \tag{40}$$

Multiplying throughout (40) by sinusoidal term on the left and substituting the expression for Q_k, we get

$$f(x) = \frac{\sin \pi \frac{x}{h}}{2\pi i} \oint_{(C_1+C_2+C_3+C_4)} \left(\frac{f(z)}{\sin\left(\pi \frac{z}{h}\right)(z - x)} \right) dz + \sum_{k \in Ind} \sin \pi \frac{x}{h} \frac{(-1)^k f(kh)}{\frac{\pi}{h}(kh - x)} \tag{41}$$

The second term in (40) corresponds to the first M terms of the infinite reconstruction sum (15). Thus, it can be inferred that the first term in (40) is the truncation error. Hence, it is true that the truncation error is given by

$$e_t(f, x, k, h, M) = \frac{\sin\left(\pi \frac{x}{h}\right)}{2\pi i} \oint_{C'} \frac{f(z)}{\sin\left(\pi \frac{z}{h}\right)(z - x)} dz$$

$$C' = C_1 + C_2 + C_3 + C_4 \tag{42}$$

The integral in (37) is computed over the contour C. By using other functions for $G(z)$, we can determine the error for various function approximations.

PROOF. To obtain a proof of Theorem 1 of Section 3.1.1, we consider each part of the C' that has a nonzero contour integral. Along the horizontal parts of the contour, C_1 and C_3, the contribution to the integral in (42) is zero. We can show this by first considering the denominator $sin(\pi z/h)$, where $z = x' + jy'$ is any point in the complex plane. It is true that

$$\left| \sin \frac{\pi(x' + jy')}{h} \right| \geq \cosh \frac{\pi}{h} |y'| \geq \sinh \frac{\pi}{h} |y'| \tag{43}$$

The numerator in (42) for all contours is bounded by $E_f cosh(\pi r \mid y' \mid /h)$ [14]. Since the *cosh* function grows faster than the *sinh* function [31], for the same argument, on contours C_1 and C_3 in the limiting case, the numerator becomes zero. Now let us consider contours C_2 and C_4. The contour C_2 lies along line $x = h(n - M - 1/2)$ and thus

$$\left| \frac{f(z)}{\sin\left(\pi \frac{z}{h}\right)(z - x)} \right| \leq \frac{E_f \cosh \frac{\pi}{h} r|y'|}{\cosh \frac{\pi}{h} |y'| \sqrt{\left(x' - h\left(n - M - \frac{1}{2}\right)\right)^2 + y'^2}} \tag{44}$$

After further simplification, the integrand along contour C_2 is now bounded by

$$\left| \frac{f(z)}{\sin\left(\pi \frac{z}{h}\right)(z - x)} \right| \leq \frac{E_f \cosh \frac{\pi}{h} ry'}{\cosh \frac{\pi}{h} y' \sqrt{(Mh)^2 + y'^2}} \tag{45}$$

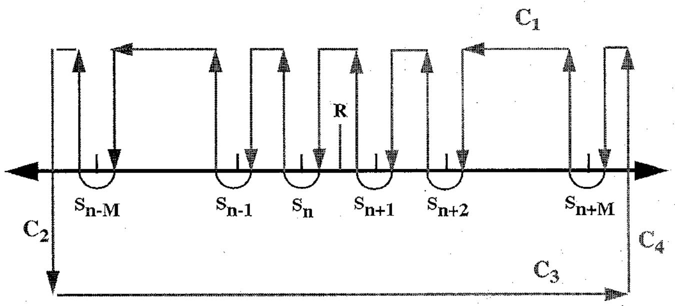


Fig. 15. Contour Integration of (40). The top portion of the contour in Fig. 14 is modified to include the poles created by the sampling points. The new contour include the segmented portion on the top (C_1), a complete horizontal path (C_3), vertical paths (C_2 and C_4), and $2M + 1$ clockwise paths leading to and from the sampled points.

It can be shown that the integrand along C_4 is also bounded by the same quantity. Recognizing that $cosh(z)$ grows faster than e^z and then evaluating the integrals for both remaining contours, we conclude that

$$e_t(f_s, x, k, h, M) \le \frac{2E_f\sqrt{\frac{r}{h}}\left|\sin\frac{\pi x}{h}\right|}{\pi^2 M(1-r)}. \quad (46)$$

ACKNOWLEDGMENTS

This work was partially supported by U.S. National Science Foundation Grant CCR-9211288 and by the Advanced Research Projects Agency Contract DABT63-C-0056. We would like to thank the following individuals at the Ohio State University—Ajeetkumar Gaddipatti and Ed Swan for reading the manuscript and for offering useful suggestions, Ashok Krishnamurthy, Department of Electrical Engineering, for suggesting that we look at spatial methods, Bhavik Bhakshi, Department of Chemical Engineering, for discussions on functional representation of sampled datasets, errors, and adaptive methods, Wayne Carlson and Steve May, Advanced Computing Center for the Arts and Design, and Don Stredney of the Ohio Supercomputer Center for the generous use of various resources and advice. Thanks also go to Tom Malzbender of HP Labs and Peter Schroeder and Jim Arvo of the California Institute of Technology for useful comments and suggestions.

REFERENCES

[1] M.J. Bentum, B.B.A. Lichtenbelt, and T. Malzbender "Frequency Analysis of Gradient Estimators in Volume Rendering," *IEEE Trans. Visualization and Computer Graphics*, vol. 2, no. 3, pp. 242-253, Sept. 1996.

[2] T. Bier and S. Neely, "Feature Based Image Metamorphosis," *Proc. SIGGRAPH '92, Computer Graphics*, vol. 26, no. 2, pp. 35-42, July 1992.

[3] I. Carlbom, "Optimal Filter Design for Volume Reconstruction and Visualization," *Proc. Visualization '93*, pp. 54-61, 1993.

[4] S.D. Casey and D.F. Walnut, "Systems of Convolution Equations, Deconvolutions, Shannon Sampling and Gabor Transforms," *SIAM Review*, vol. 36, no. 4, pp. 537-577, Dec. 1994.

[5] E. Catmull and A.R. Smith, "Three-Dimensional Transformations of Images in Scanline Order," *Computer Graphics*, vol. 14, no. 3, pp. 279-285, July 1980.

[6] R.E. Crochiere and L.R. Rabiner, *Multirate Digital Signal Processing*. Englewood Cliffs, N.J.: Prentice Hall, 1983.

[7] D.E. Dudgeon and R.M. Mersereau, *Multidimensional Digital Signal Processing*. Englewood Cliffs, N.J.: Prentice Hall, 1984.

[8] A. Fournier and E. Fiume "Constant-Time Filtering with Space-Variant Kernels," *Computer Graphics*, vol. 22, no. 4, pp. 229-238, Aug. 1988.

[9] D.M. Funderburke and S. Park, "Real-Time Pitch (Frequency) Shifting Techniques," *Proc. ICSPAT '93*, pp. 89-96.

[10] A. Glassner, *Principles of Digital Image Synthesis*. Morgan Kaufmann, 1995.

[11] M.E. Goss, "An Adjustable Gradient Filter for Volume Visualization Image Enhancement," *Proc. Graphics Interface '94*, pp. 67-74, Toronto, Canada, 1994.

[12] P. Hanrahan, "Three-Pass Affine Transforms for Volume Rendering," *Computer Graphics*, vol. 24, no. 5, pp. 71-77, Nov. 1990.

[13] P.S. Heckbert, "Survey of Texture Mapping," *IEEE Computer Graphics and Applications*, vol. 6, no. 11, pp. 56-67, 1986.

[14] H.D. Helm and J.B. Thomas, "Truncation Error of Sampling Theorems," *Proc. IRE*, vol. 50, pp. 179-184, Feb. 1962.

[15] A.K. Jain, *Fundamentals of Digital Image Processing*. Englewood Cliffs, N.J.: Prentice Hall, 1989.

[16] M. Levoy, "Display of Surfaces from Volume Data," *IEEE Computer Graphics and Applications*, vol. 8, no. 5, pp. 29-37, May 1988.

[17] *Volume Visualization*, A. Kaufman, ed. Los Alamitos, Calif.: IEEE CS Press, 1990.

[18] R.G. Keys, "Cubic Convolution Interpolation for Digital Image Processing," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 29, no. 6, pp. 1,153-1,160, Dec. 1981.

[19] Y. Kurzion and R. Yagel, "Space Deformation Using Ray Deflectors," *Proc. Sixth Eurographics Workshop on Rendering*, pp. 21-32, Dublin, June 1995.

[20] T. Malzbander, "Fourier Volume Rendering," *ACM Trans. Computer Graphics*, vol. 12, no. 3, pp. 233-250, July 1993.

[21] N. Max, "An Optimal Filter for Image Reconstruction," *Graphics Gems II*, pp. 101-104. Academic Press, 1991.

[22] R. Machiraju, E. Swan, and R. Yagel, "Spatial Domain Characterization and Control of Reconstruction Errors," *Proc. Sixth Eurographics Workshop on Rendering*, pp. 64-73, Dublin, June 1995.

[23] S.R. Marschner and R.J. Lobb, "An Evaluation of Reconstruction Filters for Volume Rendering," *Proc. Visualization '94*, pp. 100-107, Oct. 1994.

[24] D.P. Mitchell and A.N. Netravali, "Reconstruction Filters in Computer Graphics," *Computer Graphics*, vol. 22, no. 4, pp. 221-228, Aug. 1988.

[25] A. Norton, A.P. Rockwood, and P.T. Skolmoski, "Clamping: A Method of Antialiasing Textured Surfaces by Bandwidth Limiting in Object Space," *Proc. SIGGRAPH '82, Computer Graphics*, vol. 16, no. 3, pp. 1-8, July 1992.

[26] A.V. Oppenheim and R.W. Schafer, *Digital Signal Processing*. Englewoods Cliffs, N.J.: Prentice Hall, 1975.

[27] S.K. Park and R.A. Schowengerdt, "Image Reconstruction by Parametric Cubic Convolution," *Computer Vision, Graphics, and Image Processing*, vol. 23, pp. 258-272, 1983.

[28] A. Paeth, "A Fast Algorithm for General Raster Rotation," *Proc. Graphics Interface '86*, pp. 77-81, May 1986.

[29] J.A. Parker, R.V. Kenyon, and D.E. Troxel, "Comparison of Interpolation Methods for Image Resampling," *IEEE Trans. Medical Imaging*, vol. 2, no. 1, pp. 31-39, Mar. 1983.

[30] M. Rhodes, W. Glenn, and Y. Azzawi, "Extracting Oblique Planes from Serial CT Sections," *J. Computer Assisted Tomography*, vol. 4, no. 5, pp. 649-657, Oct. 1980.

[31] E.B. Saff and A.D. Snider, *Fundamentals of Complex Analysis for Mathematics, Science, and Engineering*. Englewood Cliffs, N.J.: Prentice Hall, 1976.

[32] W. Schreiber and D. Troxel, "Transformation Between Continuous and Discrete Representations of Images: A Perceptual Approach," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 7, no. 2, pp. 178-186, Mar. 1985.

[33] F. Stenger, *Numerical Methods Based on Sinc and Analytic Functions*. Springer Verlag, 1993.

[34] R. Sudhakar, R.C. Agarwal, and S.C. Dutta Roy, "Time Domain Interpolation Using Differentiators," *IEEE Trans. Accoustics, Speech, and Signal Processing*, vol. 30, no. 6, pp. 992-997, Dec. 1982.

[35] T. Totsuka and M. Levoy, "Frequency Domain Volume Rendering," *Computer Graphics Proc. SIGGRAPH '93*, pp. 271-278, Aug. 1993.

[36] K. Turkowski, "Filters for Common Resampling Tasks," *Graphics Gems I*, pp. 147-165. Academic Press, 1991.

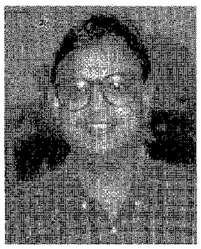
[37] L. Westover, "Footprint Evaluation for Volume Rendering," *Computer Graphics*, vol. 24, no. 4, pp. 367-376, Aug. 1990.

[38] G. Wolberg, *Digital Image Warping*. Los Alamitos, Calif.: IEEE CS Press, 1990.

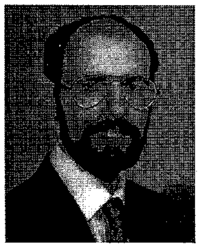
[39] R. Yagel and A.E. Kaufman, "Template-Based Volume Viewing," *Computer Graphics Forum*, vol. 11, no. 3, pp. 153-157, Sept. 1992.

[40] K. Yao and J.B. Thomas, "On Truncation Error Bounds for Sampling Representations of Band-Limited Signals," *IEEE Trans. Aerospace and Electronic Systems*, vol. 2, no. 6, pp. 640-647, Nov. 1966.

[41] A.I. Zayed, *Advances in Shannon's Sampling Theory*. Boca Raton, Fla.: CRC Press, 1993.



Raghu Machiraju received his BSc from the University of Delhi, India, in 1982, his ME in automation from the Indian Institute of Science, Bangalore, India, in 1984, and his PhD in 1996 from the Ohio State University. Before attending Ohio State, he worked for the Ohio Supercomputer Center and Control Data Corporation. Dr. Machiraju is now an assistant professor in the Department of Computer Science at Mississippi State University. His current research interests are in accurate and adaptive methods for volume rendering and wavelet approach to feature-based image and volume processing.



Roni Yagel received his BSc cum laude and his MSc cum laude from the Department of Mathematics and Computer Science at Ben Gurion University, Israel, in 1986 and 1987, respectively, and his PhD in 1991 from the State University of New York at Stony Brook. He was also a researcher in the Department of Anatomy and the Department of Physiology and Biophysics at SUNY Stony Brook. Dr. Yagel is now an assistant professor in the Department of Computer and Information Science at the Ohio State University. His research and technical publications deal with various aspects of volume rendering. His research interests also include algorithms for volume graphics, imaging, animation, and visualization tools for surgery simulation and other biomedical applications.